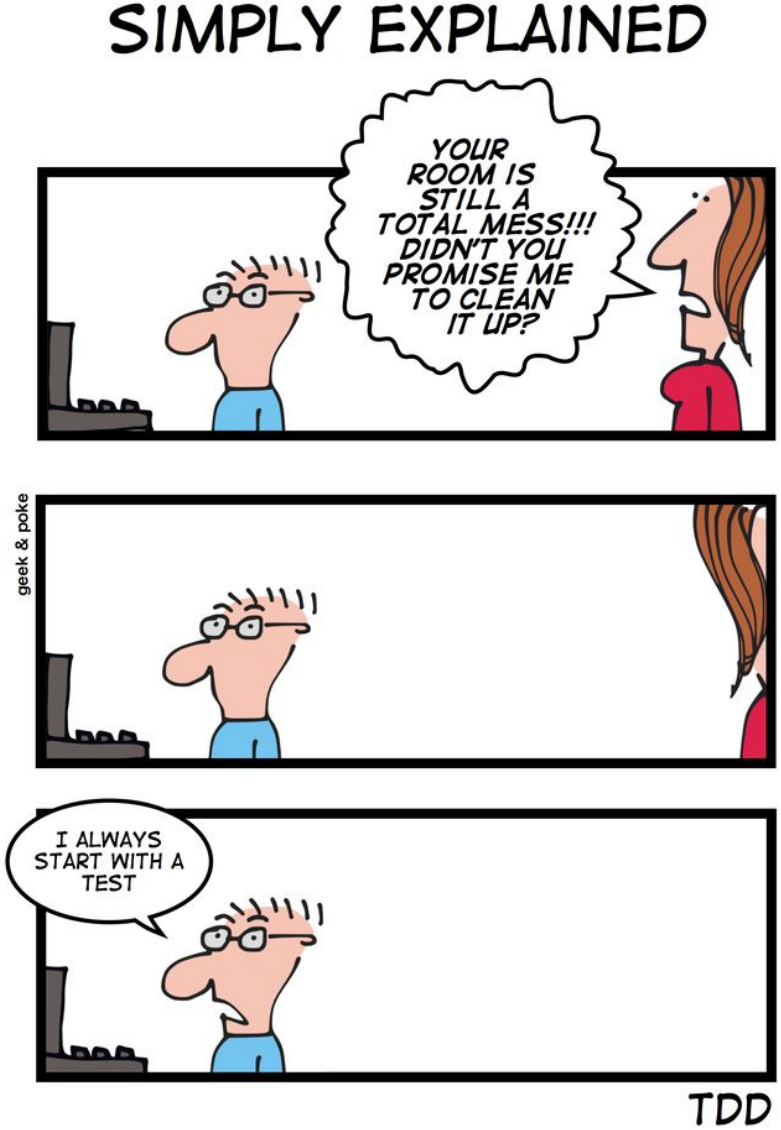

SIMPLY EXPLAINED
YOUR ROOM IS STILL A TOTAL MESS!!! DIDN'T YOU PROMISE ME TO CLEAN IT UP?
geek & poke
I ALWAYS START WITH A TEST
TDD

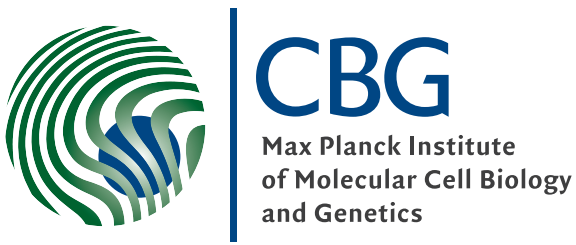
Unit Tests and Test-Driven Development

- An Introduction -

Peter Steinbach

Scionics Computer Innovations GmbH

October 8th, 2011



Outline

Unit Testing

Unit Testing

Definition

A method by which individual units of source code are tested to determine if they are fit for use. A unit is the smallest testable part of an application. (from [2])

Unit Testing

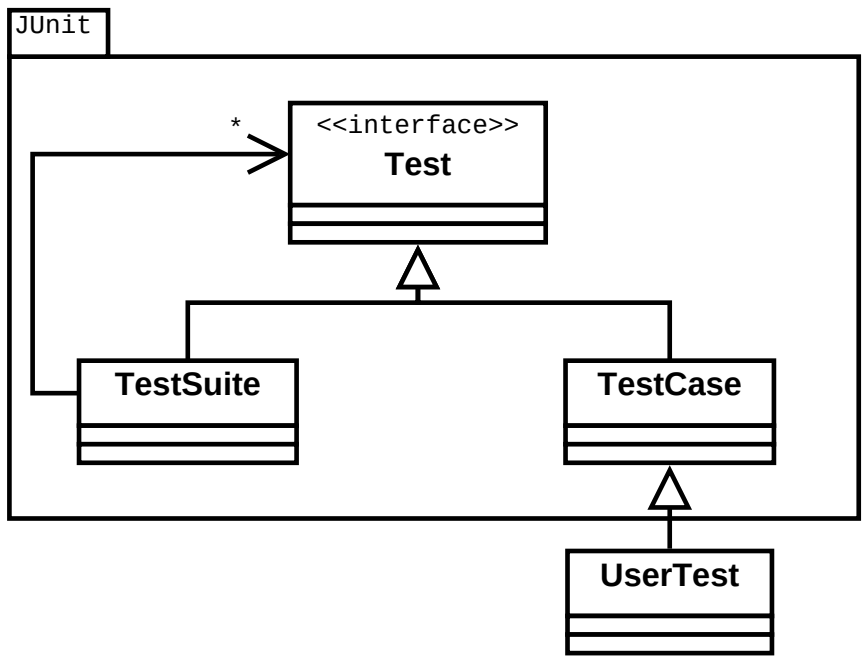
Definition

A method by which individual units of source code are tested to determine if they are fit for use. A unit is the smallest testable part of an application. (from [2])

Where did it come from?

- `JUnit` first unit test framework in java (written by Kent Beck and Erich Gamma)
- since then, many ports to other languages written (see [4] and [1])
- provides unified interface and clean environment
- for `C++`, I prefer boost's unified testing framework [3]
- for `python`, I plainly use the `unittest` module [5]

How does it work?



Unit Testing: An Example : The MagVector class

Design Goals

- class containing a set of floats (in view of an algebraic vector)

Unit Testing: An Example : The MagVector class

Design Goals

- class containing a set of floats (in view of an algebraic vector)
- class able to compute the norm (or metric) of the vector

Unit Testing: An Example : The MagVector class

Design Goals

- class containing a set of floats (in view of an algebraic vector)
- class able to compute the norm (or metric) of the vector

Let's have a look at the implementation of the **MagVector class and its tests** on your exercise sheets!

Unit Testing: Take-Home Messages

What to test?

1. contract of a class (its requirements and responsibilities)

Unit Testing: Take-Home Messages

What to test?

1. contract of a class (its requirements and responsibilities)
2. how exceptions/errors are handled (corner cases)

Unit Testing: Take-Home Messages

What to test?

1. contract of a class (its requirements and responsibilities)
2. how exceptions/errors are handled (corner cases)

Unit Testing: Take-Home Messages

What to test?

1. contract of a class (its requirements and responsibilities)
2. how exceptions/errors are handled (corner cases)

Essentials about unit testing

- important that tests run/compile quick and immediately
- provides **fast** feedback to developer
- good IDEs have plugins that make testing very easy

Testing in Software Development

Testing in Software Development: Where Unit Tests are used

Tests are conducted on the unit, module or system level usually by dedicated Test or Q&A Engineers!

Testing in Software Development: Where Unit Tests are used

Tests are conducted on the unit, module or system level usually by dedicated Test or Q&A Engineers!

static testing

- code is not executed
- code metrics, code reviews

dynamic testing

- code is executed
- input/output is checked

Testing in Software Development: Where Unit Tests are used

Tests are conducted on the unit, module or system level usually by dedicated Test or Q&A Engineers!

static testing

- code is not executed
- code metrics, code reviews

dynamic testing

- code is executed
- input/output is checked

white box testing

- software internals are analysed
- execution paths are checked

black box testing

- client perspective to software (internals assumed unknown)
- valid/invalid inputs given

Testing in Software Development: Where Unit Tests are used

Tests are conducted on the unit, module or system level usually by dedicated Test or Q&A Engineers!

static testing

- code is not executed
- code metrics, code reviews

dynamic testing

- code is executed
- input/output is checked

white box testing

- software internals are analysed
- execution paths are checked

black box testing

- client perspective to software (internals assumed unknown)
- valid/invalid inputs given

. . . and many more!

Testing in Software Development: Where Unit Tests are used

Tests are conducted on the unit, module or system level usually by dedicated Test or Q&A Engineers!

static testing

- code is not executed
- code metrics, code reviews

dynamic testing

- code is executed
- input/output is checked

white box testing

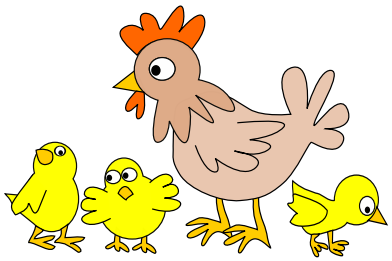
- software internals are analysed
- execution paths are checked

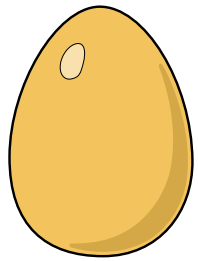
black box testing

- client perspective to software (internals assumed unknown)
- valid/invalid inputs given

. . . and many more!

Testing in Software Development: Most Common Applications to Unit Tests



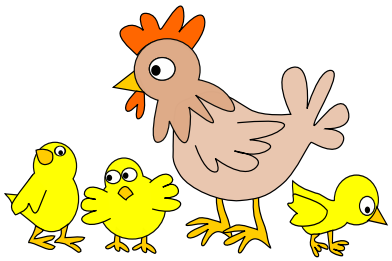


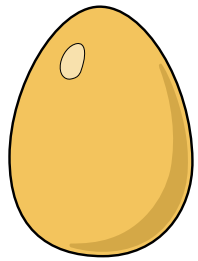
A-Postiori Tests

- write tests to check if implementation meets design intentions

A-Priori Tests

Testing in Software Development: Most Common Applications to Unit Tests



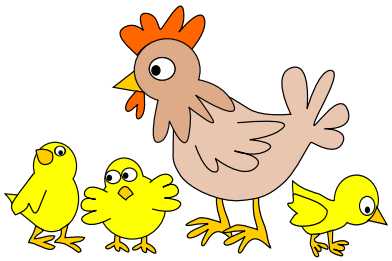


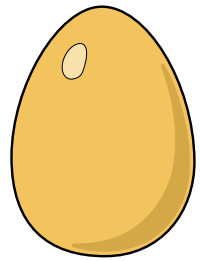
A-Postiori Tests

- write tests to check if implementation meets design intentions
- write (one) test(s) for each bug report

A-Priori Tests

Testing in Software Development: Most Common Applications to Unit Tests



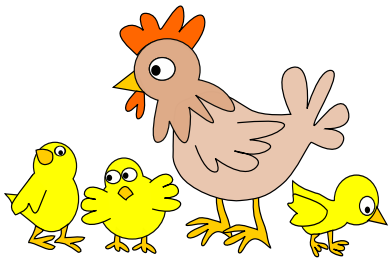


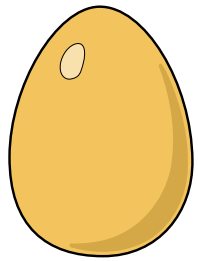
A-Postiori Tests

- write tests to check if implementation meets design intentions
- write (one) test(s) for each bug report
- write tests to fix performance

A-Priori Tests

Testing in Software Development: Most Common Applications to Unit Tests



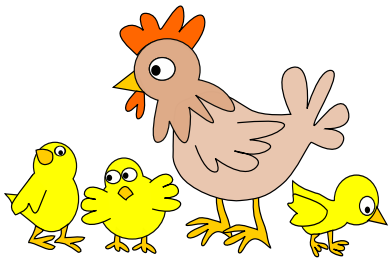


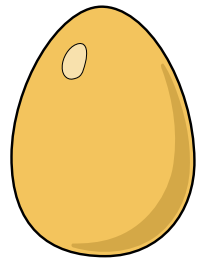
A-Postiori Tests

- write tests to check if implementation meets design intentions
- write (one) test(s) for each bug report
- write tests to fix performance
- write tests to fix behavior (refactoring legacy code)

A-Priori Tests

Testing in Software Development: Most Common Applications to Unit Tests





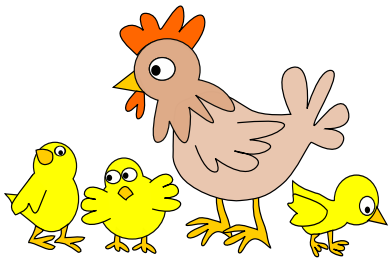
A-Postiori Tests

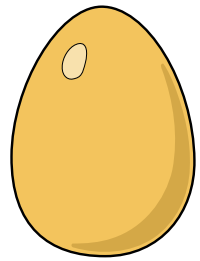
- write tests to check if implementation meets design intentions
- write (one) test(s) for each bug report
- write tests to fix performance
- write tests to fix behavior (refactoring legacy code)

A-Priori Tests

- test first

Testing in Software Development: Most Common Applications to Unit Tests





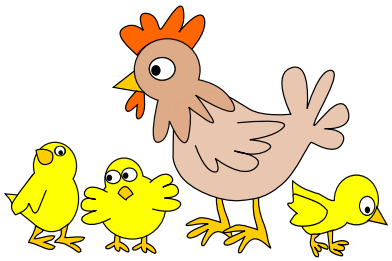
A-Postiori Tests

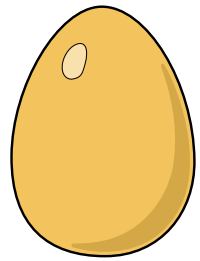
- write tests to check if implementation meets design intentions
- write (one) test(s) for each bug report
- write tests to fix performance
- write tests to fix behavior (refactoring legacy code)

A-Priori Tests

- test first
- then code

Testing in Software Development: Most Common Applications to Unit Tests





A-Postiori Tests

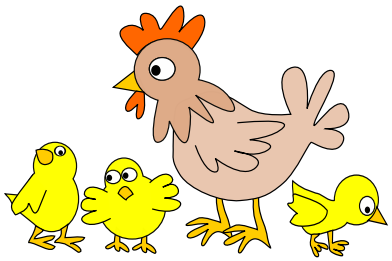
- write tests to check if implementation meets design intentions
- write (one) test(s) for each bug report
- write tests to fix performance
- write tests to fix behavior (refactoring legacy code)

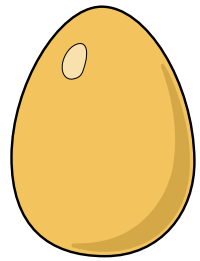
A-Priori Tests

- test first
- then code

Test-Driven Development

Testing in Software Development: Most Common Applications to Unit Tests





A-Postiori Tests

- write tests to check if implementation meets design intentions
- write (one) test(s) for each bug report
- write tests to fix performance
- write tests to fix behavior (refactoring legacy code)

A-Priori Tests

- test first
- then code

Test-Driven Development

Hen-and-egg discussion going on, [8, 9, 10].

Test-Driven Development

Test-Driven Development: Motivation

Wait ... Design-Driven not Test-Driven!

- **Yes**, before one codes a package, one should sit down with pen and paper
- **No** pen-and-paper design ends up 1:1 in production releases!

Test-Driven Development: Motivation

Wait ... Design-Driven not Test-Driven!

- **Yes**, before one codes a package, one should sit down with pen and paper
- **No** pen-and-paper design ends up 1:1 in production releases!

Change is everywhere

- requirements change
  ‘‘Can your package read XML files as well?’’

Test-Driven Development: Motivation

Wait ... Design-Driven not Test-Driven!

- **Yes**, before one codes a package, one should sit down with pen and paper
- **No** pen-and-paper design ends up 1:1 in production releases!

Change is everywhere

- requirements change
  ‘‘Can your package read XML files as well?’’
- environments change
  ‘‘We are moving from std::vector to tbb::concurrent_vector! Please provide a check-in until tomorrow!’’

Test-Driven Development: Motivation

Wait ... Design-Driven not Test-Driven!

- **Yes**, before one codes a package, one should sit down with pen and paper
- **No** pen-and-paper design ends up 1:1 in production releases!

Change is everywhere

- requirements change
  ‘‘Can your package read XML files as well?’’
- environments change
  ‘‘We are moving from std::vector to tbb::concurrent_vector! Please provide a check-in until tomorrow!’’
- experience grows
  ‘‘Why in the world did I ever write such crap?’’

Test-Driven Development: Motivation

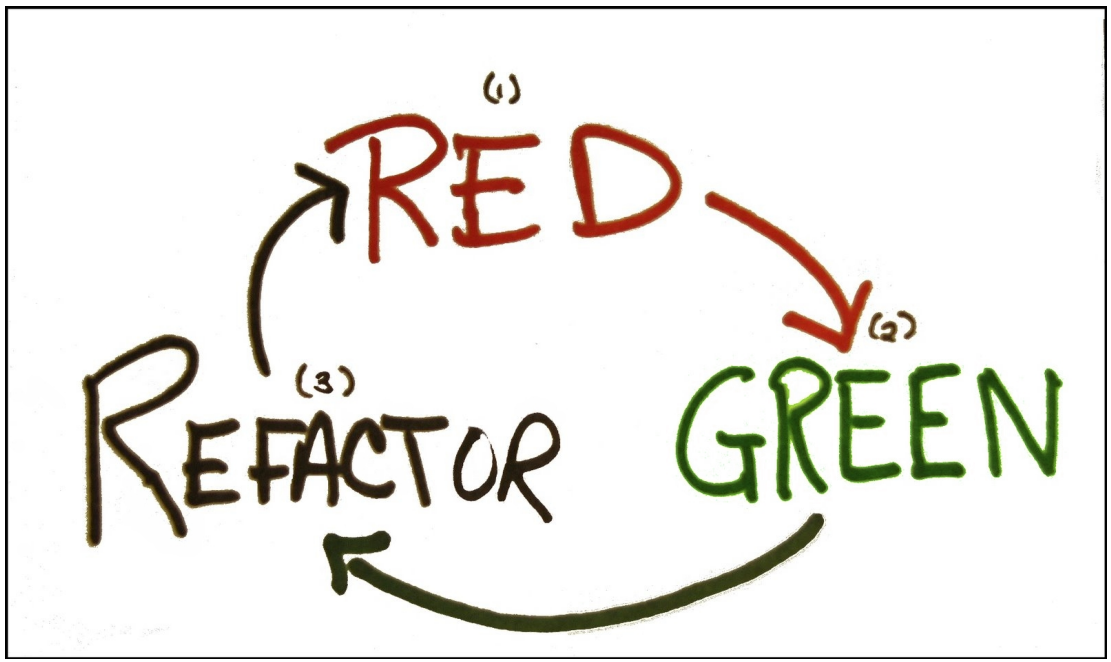
Wait ... Design-Driven not Test-Driven!

- **Yes**, before one codes a package, one should sit down with pen and paper
- **No** pen-and-paper design ends up 1:1 in production releases!

Change is everywhere

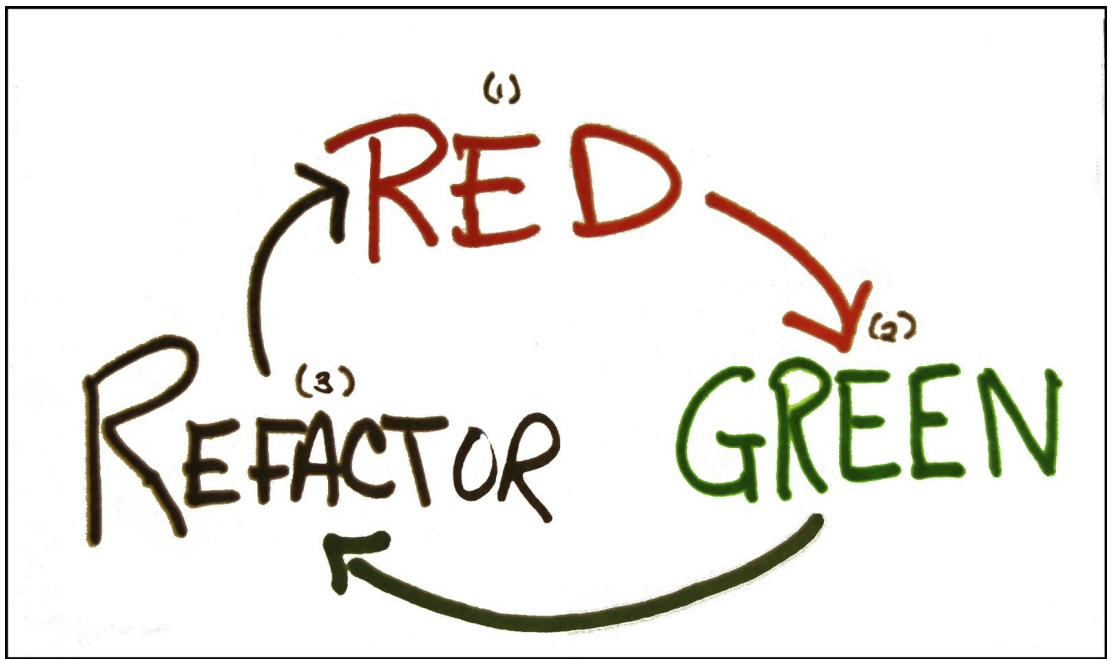
- requirements change
  ‘‘Can your package read XML files as well?’’
- environments change
  ‘‘We are moving from std::vector to tbb::concurrent_vector! Please provide a check-in until tomorrow!’’
- experience grows
  ‘‘Why in the world did I ever write such crap?’’
- upgrade legacy code
  ‘‘Ahh, and here is the code of your predecessor. You can use it as a starting point!’’

Test-Driven Development: How TDD works



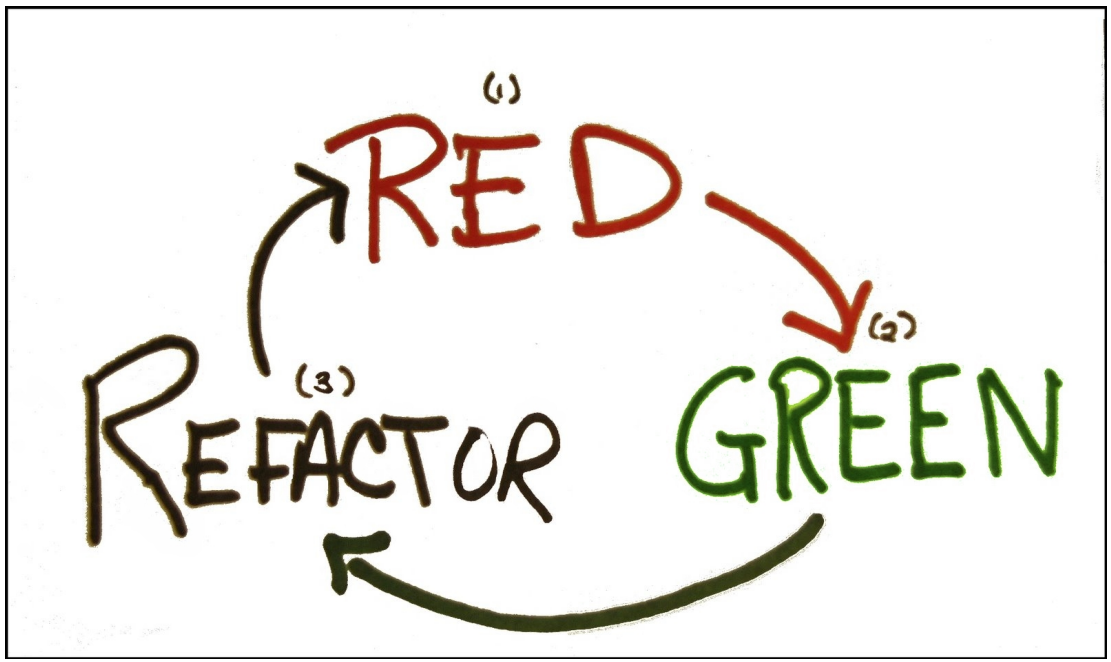
1. **write a test that fails**

Test-Driven Development: How TDD works



1. **write a test that fails**
2. **implement code that makes your test pass**

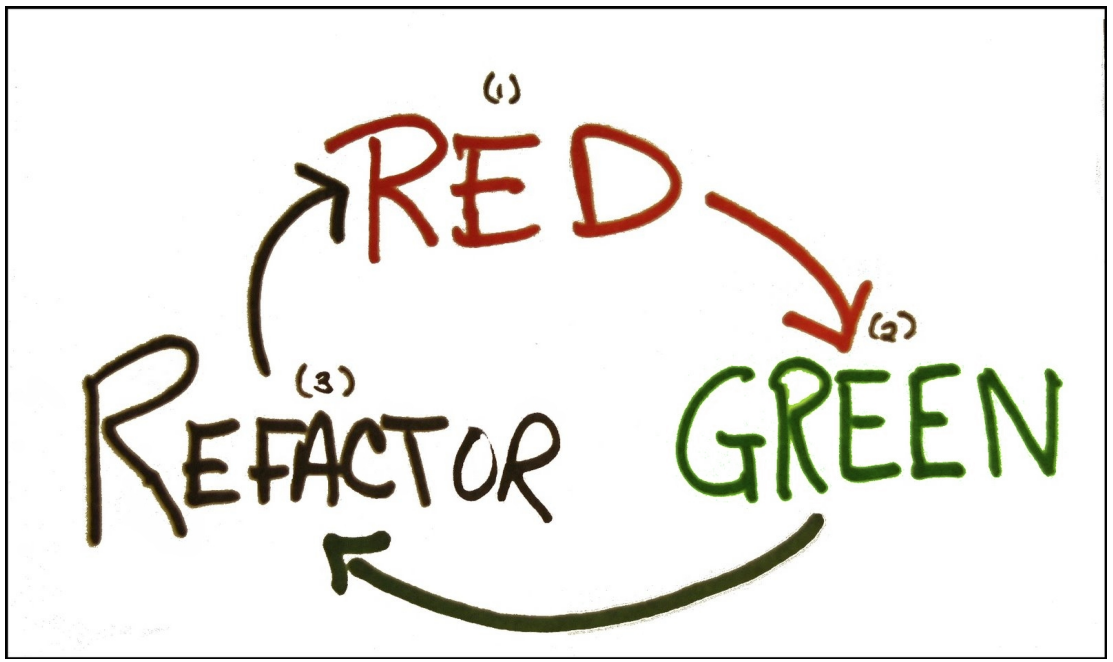
Test-Driven Development: How TDD works



1. **write a test that fails**
2. **implement code that makes your test pass**
3. **refactor your code**

   (see talk by Maria tomorrow)

Test-Driven Development: How TDD works



1. **write a test that fails**
2. **implement code that makes your test pass**
3. **refactor your code**

   (see talk by Maria tomorrow)

Background

- agile development technique [6]
- formulated by Kent Beck in 2002 ([7])
- enforces simple design, testability and treats bugs before they happen

Test-Driven Development: Demonstrate TDD

A Demonstration Might Save a 1000 Words!

Adding different norms to `MagVector`!

Test-Driven Development: What it may provide

- Better Code
  (less bugs)

Test-Driven Development: What it may provide

- Better Code
  (less bugs)
- Easier Coding
  (focus on one problem at a time)

Test-Driven Development: What it may provide

- Better Code
  (less bugs)
- Easier Coding
  (focus on one problem at a time)
- Solution Triangulation
  (iterating towards the final goal)

Test-Driven Development: What it may provide

- Better Code
  (less bugs)
- Easier Coding
  (focus on one problem at a time)
- Solution Triangulation
  (iterating towards the final goal)
- Coding as a client

Test-Driven Development: What it may provide

- Better Code
  (less bugs)
- Easier Coding
  (focus on one problem at a time)
- Solution Triangulation
  (iterating towards the final goal)
- Coding as a client
- More time thinking up front
  (clear goal is essential)

Test-Driven Development: What it may provide

- Better Code
  (less bugs)
- Easier Coding
  (focus on one problem at a time)
- Solution Triangulation
  (iterating towards the final goal)
- Coding as a client
- More time thinking up front
  (clear goal is essential)
- Better unit test coverage

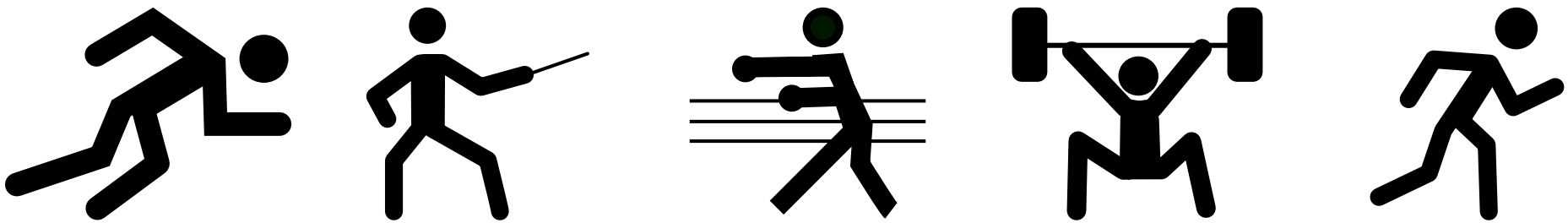
Test-Driven Development: What it may provide

- Better Code
  (less bugs)
- Easier Coding
  (focus on one problem at a time)
- Solution Triangulation
  (iterating towards the final goal)
- Coding as a client
- More time thinking up front
  (clear goal is essential)
- Better unit test coverage
- Tests are less likely to be dropped

Test-Driven Development: What it may provide

- Better Code
  (less bugs)
- Easier Coding
  (focus on one problem at a time)
- Solution Triangulation
  (iterating towards the final goal)
- Coding as a client
- More time thinking up front
  (clear goal is essential)
- Better unit test coverage
- Tests are less likely to be dropped
- no extra code
  (code the minimum required)

Test-Driven Development: Bottom Line for Every-Day Coding



Try it out! Practise is essential!

Summary

Summary

Unit Tests

- unit tests cover the behavior of the smallest entity of code (function/method/class)

Summary

Unit Tests

- unit tests cover the behavior of the smallest entity of code (function/method/class)
- a lot of unit test frameworks available for (any) popular language

Summary

Unit Tests

- unit tests cover the behavior of the smallest entity of code (function/method/class)
- a lot of unit test frameworks available for (any) popular language

Summary

Unit Tests

- unit tests cover the behavior of the smallest entity of code (function/method/class)
- a lot of unit test frameworks available for (any) popular language

TDD ...

- is a coding method to create tests and production code at the same time

Summary

Unit Tests

- unit tests cover the behavior of the smallest entity of code (function/method/class)
- a lot of unit test frameworks available for (any) popular language

TDD . . .

- is a coding method to create tests and production code at the same time
- can provide good quality code that is tested

Summary

Unit Tests

- unit tests cover the behavior of the smallest entity of code (function/method/class)
- a lot of unit test frameworks available for (any) popular language

TDD . . .

- is a coding method to create tests and production code at the same time
- can provide good quality code that is tested
- should not be applied blindly

Summary

Unit Tests

- unit tests cover the behavior of the smallest entity of code (function/method/class)
- a lot of unit test frameworks available for (any) popular language

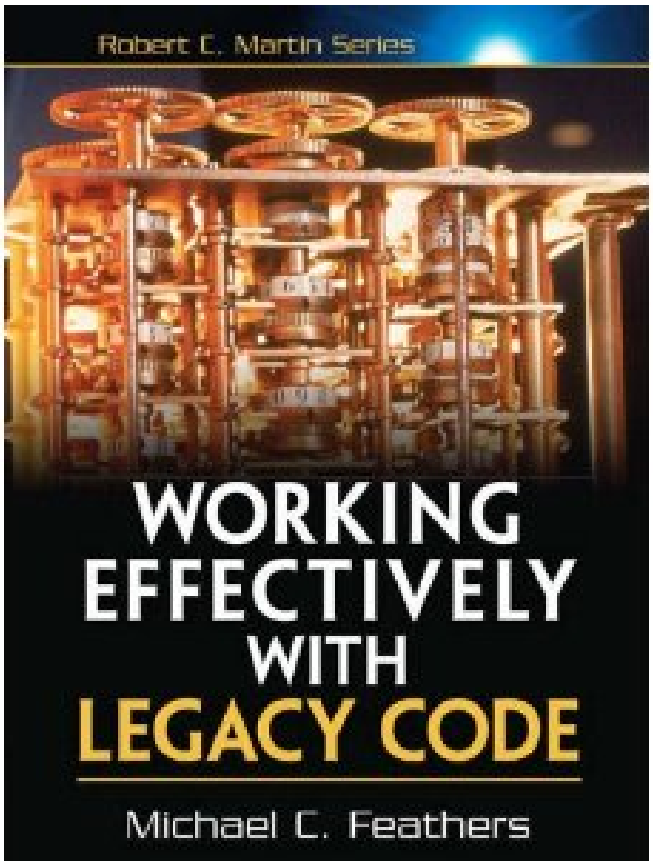
TDD ...

- is a coding method to create tests and production code at the same time
- can provide good quality code that is tested
- should not be applied blindly
- needs practise

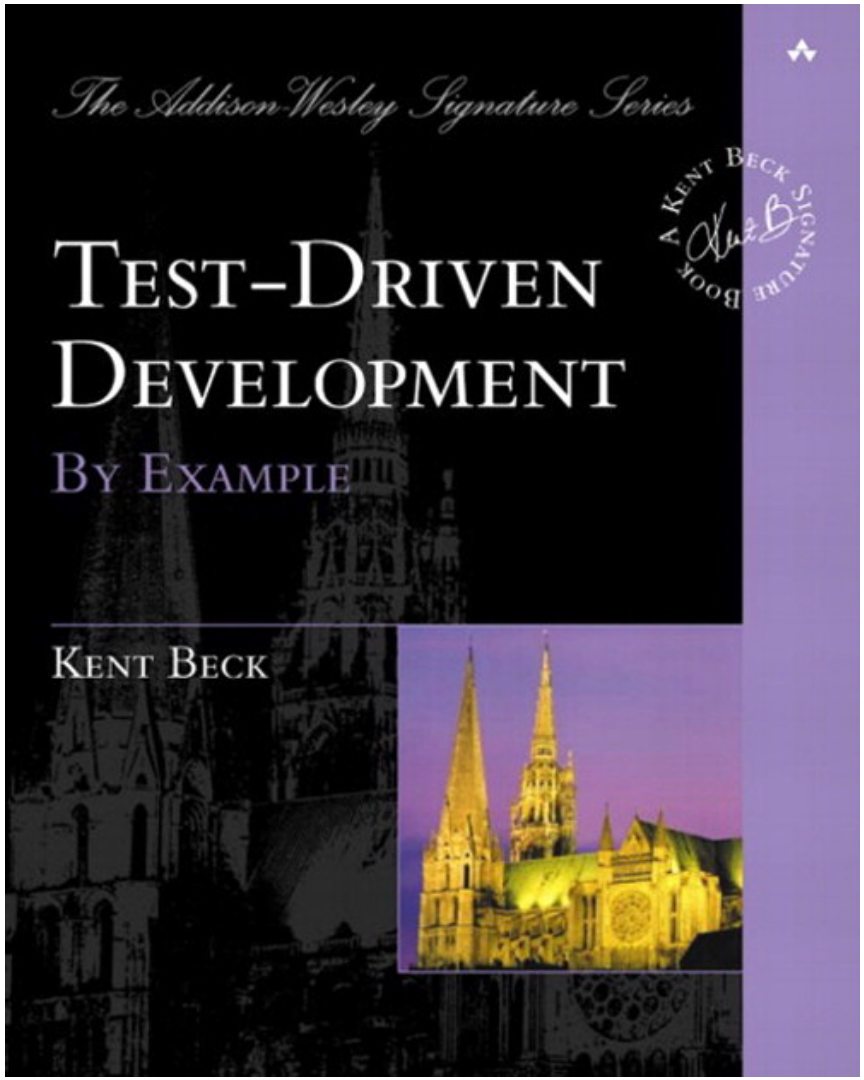
Try *TDD* to see yourself!

Thank you for your attention!

Literature



[11]



[7]

References

[1] http://en.wikipedia.org/wiki/list_of_unit_testing_frameworks.

[2] http://en.wikipedia.org/wiki/unit_tests.

[3] http://www.boost.org/doc/libs/1_51_0/libs/test/doc/html/utf.html.

[4] http://www.xprogramming.com/software.htm.

[5] Unittest module documentation.
docs.python.org.
docs.python.org/library/unittest.html.

[6] Agile manifesto.
web, 2001.
agilemanifesto.org.

[7] Kent Beck.
Test-Driven Development by Example.
Number ISBN 0321146530. Addison-Wesley Longman, 2002.

[8] Cedric Beust.
Breaking away from the unit test group think.
Dr. Dobbs Online, 2011.
http://drdobbs.com/architecture-and-design/231600404.

[9] Andrew Binstock.
Unit testing: Is there really any debate any longer?
Dr. Dobbs Online, 2012.
http://www.drdobbs.com/testing/unit-testing-is-there-really-any-debate/240007176.

[10] Joe Eames.
Tdd: Is there really any debate any longer?
Dr. Dobbs Online, 2012.
http://www.drdobbs.com/testing/tdd-is-there-really-any-debate-any-longe/240007457.

[11] Michael Feathers.
Working Effectively with Legacy Code.
Robert C. Martin Series. Prentice Hall, 2004.